**2312647\_Lê Anh Khoa\_CTK47A**

**LAB 2**

# I. Mục đích

Ban đầu làm quen với lập trình Socket

Hiểu được cách hoạt động của bộ đệm hệ thống và bộ đệm chương trình

Xây dựng chương trình client-server đơn giản

**II. Yêu cầu**

Xây dựng các chương trình client-server từ mức đơn giản nhất rồi phức tạp dần

# III. Hướng dẫn

## III.1. Tạo một chương trình server đơn giản

### III.1.1. Hướng dẫn lập trình

Để lập trình socket ta sử dụng 2 namespace:

using System.Net;

using System.Net.Sockets;

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Any, 5000);

Tạo Server Socket, Socket này sẽ được kết nối với Server EndPoint:

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,

ProtocolType.Tcp);

Lắng nghe kết nối trên Server Socket, khi có client kết nối tới sẽ trả về Socket để trao đổi dữ liệu với client:

serverSocket.Listen(10);

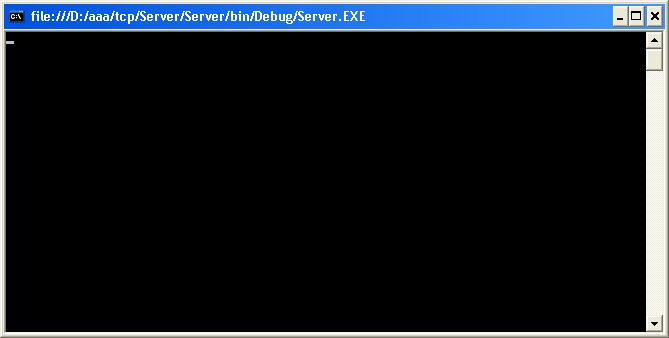
Socket clientSocket = serverSocket.Accept();

Khi client kết nối tới trên server sẽ xuất thông tin của client kết nối tới (địa chỉ IP, Port)

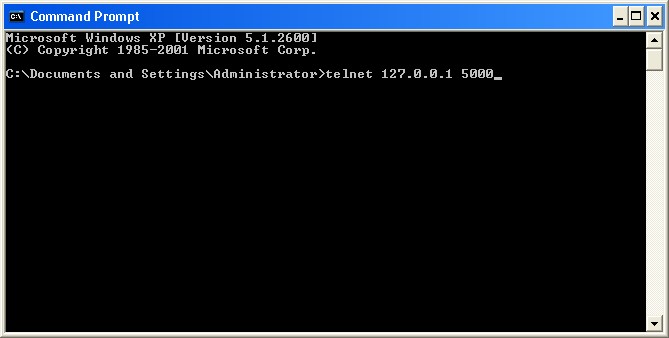
EndPoint clientEndPoint = clientSocket.RemoteEndPoint; Console.WriteLine(clientEndPoint.ToString());

**III.1.2. Test chương trình:**

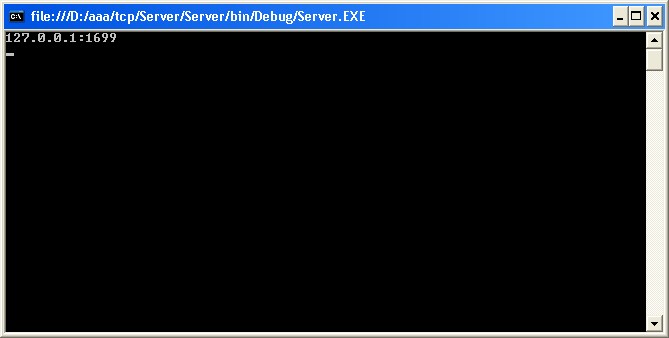
Chạy chương trình Server:



Chạy chương trình CMD và telnet vào server trên với địa chỉ IP loopback và port 5000:



Kết quả trả về trên Server:



**III.1.3. Trả lời câu hỏi:**

1. **Khi telnet vào Server bằng port 5000 tại sao port client dùng để kết nối vào lại không phải là port 5000 ?**

* Vì client kết nối đến, hệ điều hành sẽ tự động cấp phát một port ngẫu nhiên cho client để thực hiện kết nối. Port này khác với 5000 để tránh xung đột (nếu client cũng dùng 5000 thì sẽ trùng với server).

1. **Như hình trên, giải thích vì sao có client dùng port giá trị 1699 ?**

* Con số 1699 là một port ngẫu nhiên mà hệ điều hành gán cho tiến trình client tại thời điểm đó. Mỗi lần client mở kết nối, OS chọn một port tự do trong khoảng 1024-65535. Vì vậy nó có thể là 1699, 50951, 30000... tùy lúc.

1. **Có phải lúc nào client cũng mở port 1699 để kết nối với Server không ?**

* Port client luôn thay đổi, mỗi lần chạy sẽ khác nhau. Nó phụ thuộc vào OS chọn ngẫu nhiên trong dải port ngẫu nhiên. Vì vậy không cố định 1699.

1. **Ta có thể chạy 2 chương trình CMD để telnet vào Server trên được không ? Giải thích.**

* Server đã Listen(10) nghĩa là nó chấp nhận tối đa 10 kết nối đồng thời trong hàng đợi. Mỗi client kết nối sẽ được cấp một socket riêng với một port client riêng. Do đó có thể mở 2 CMD (hoặc nhiều hơn) để telnet đến cùng một server port 5000.

## III.2. Tạo chương trình Server đơn giản, khi Client kết nối vào thì gởi câu chào cho client

### III.2.1. Hướng dẫn lập trình

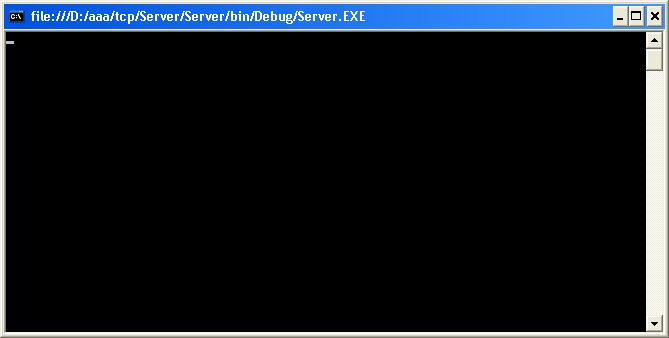
Sử dụng chương trình Server đơn giản ở trên, thêm phần gởi dữ liệu xuống client

byte[] buff;

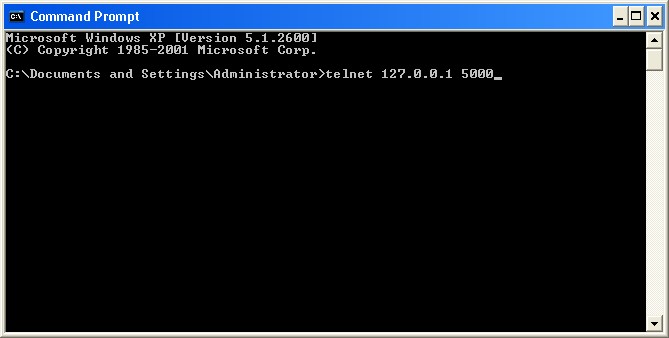
string hello = "Hello Client"; buff = Encoding.ASCII.GetBytes(hello); clientSocket.Send(buff, 0, buff.Length, SocketFlags.None);

**III.2.2. Test chương trình**

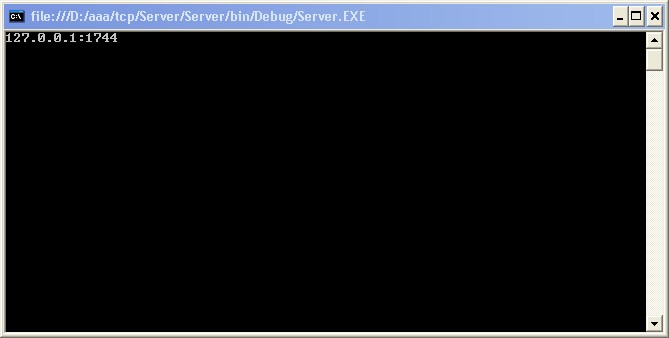
Chạy chương trình Server:



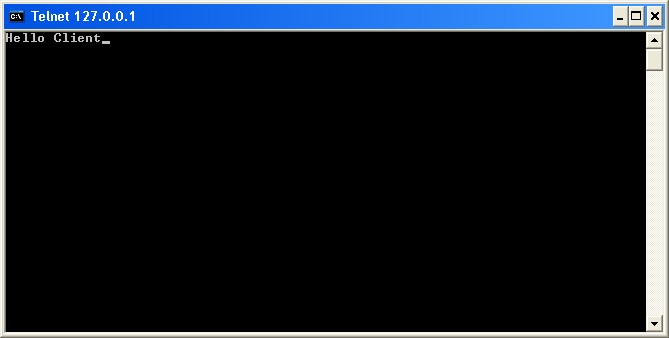
Chạy chương trình CMD và telnet vào server trên với địa chỉ IP loopback và port 5000:



Kết quả trả về trên Server:



Kết quả trả về trên client :



**III.2.3. Trả lời câu hỏi:**

1. **Tạo sao trong đoạn code viết thêm vào ta không cần phải khởi tạo độ lớn ban đầu cho buffer ?**

* Encoding.ASCII.GetBytes(hello) sẽ tự động tạo ra một mảng byte đúng bằng độ dài chuỗi "Hello Client".Do đó không cần phải khai báo trước buff = new byte[1024]; như khi làm việc với dữ liệu nhận (Receive).

1. **Giải thích ý nghĩa các tham số của phương thức clientSocket.Send .**

* buff: mảng type chứa dữ liệu cần gửi. 0: vị trí bắt đầu trong mảng buff (offset), buff.Length: số byte cần gửi (độ dài của dữ liệu). SocketFlags.None: cờ điều khiển việc gửi. None nghĩa là không dùng cờ đặc biệt (gửi bình thường).

## III.3. Xây dựng chương trình client đơn giản

### III.3.1. Hướng dẫn lập trình

Để lập trình socket ta using 2 namespace:

using System.Net;

using System.Net.Sockets;

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Loopback, 5000);

Tạo socket dùng để kết nối với server:

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,

ProtocolType.Tcp);

Console.WriteLine("Dang ket noi voi server..."); serverSocket.Connect(serverEndPoint);

Nếu kết nối thành công, nhận câu chào từ server và hiển thị câu chào lên màn hình:

if (serverSocket.Connected)

{

Console.WriteLine("Ket noi thanh cong voi server ...");

byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

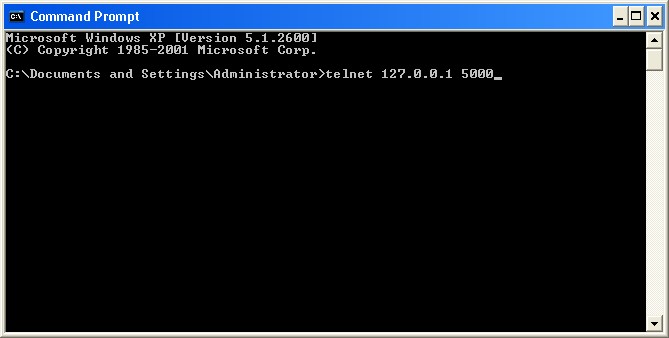
str = Encoding.ASCII.GetString(buff, 0, byteReceive)

Console.WriteLine(str);

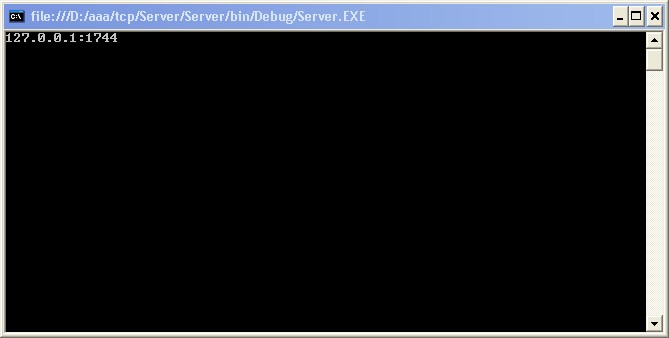
}

**III.3.2. Test chương trình**

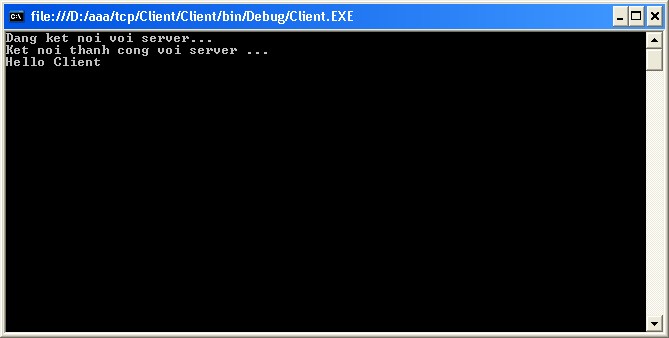
Chạy chương trình Server:



Kết quả trả về trên Server:



Kết quả trả về trên client :



**III.3.3. Trả lời câu hỏi:**

1. **Khi client kết nối đến server mà server chưa chạy hoạc là đường truyền mạng bị rớt thì có hiện tượng gì xảy ra ?, giải thích.**

* Hiện tượng: Client sẽ **không kết nối được tới server**.
* Trong chương trình C#, lệnh clientSocket.Connect(serverEndPoint) sẽ ném ra ngoại lệ SocketException với thông báo kiểu như:

No connection could be made because the target machine actively refused it 127.0.0.1:5000

* Nguyên nhân:
* Khi server chưa chạy, cổng 5000 chưa được tiến trình nào lắng nghe → client gửi yêu cầu kết nối thì hệ điều hành từ chối.
* Khi đường truyền bị rớt, gói tin SYN không đến được server → sau một khoảng thời gian client sẽ báo lỗi timeout hoặc connection refused.

1. **Từ đoạn code ở chương trình client:**

str = Encoding.ASCII.GetString(buff, 0, byteReceive)

**Nếu ta thay byteReceive bằng buff.Lengh thì kết quả sẽ như thế nào? Giải thích.**

* **Đúng:** Sử dụng byteReceive (số byte thực tế nhận được). → Chỉ chuyển đổi đúng số ký tự server gửi (ví dụ: "Hello Client").
* **Sai:** Nếu thay bằng buff.Length (kích thước bộ đệm, ví dụ 1024). → Chuỗi kết quả sẽ có "Hello Client" **kèm theo nhiều ký tự rác hoặc ký tự null (\0)** phía sau.
* Giải thích: Bộ đệm buff luôn có kích thước cố định, nhưng server chỉ gửi một phần trong đó. Nếu lấy hết buff.Length, ta sẽ giải mã cả vùng chưa có dữ liệu hợp lệ.

Ví dụ:  
Server gửi "Hello" (5 byte), nhưng buff.Length = 1024.  
→ Kết quả chuỗi sẽ là "Hello" + 1019 ký tự trống.

## III.4. Cải tiến chương trình client

### III.4.1. Hướng dẫn lập trình

Chương trình client này khi kết nối đến server mà server chưa bật hoạc đường truyền bị lỗi thì chương trình sẽ bị lỗi.

Sửa lại chương trình bằng cách thêm try-catch vào:

try {

serverSocket.Connect(serverEndPoint);

}

catch (SocketException se)

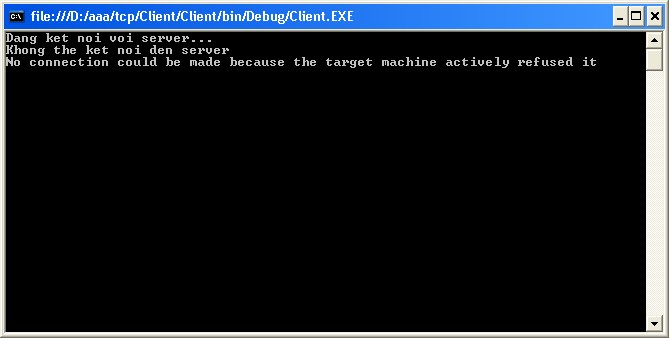
{

Console.WriteLine("Khong the ket noi den server"); return;

}

### III.4.2. Test chương trình

Chạy chương trình ở client mà không chạy server:



## III.5. Thêm khả năng gởi dữ liệu lên server và nhận lại dữ liệu từ server gởi về

### III.5.1. Hướng dẫn lập trình

Việc này được thực hiện bằng cách thêm vòng lặp vô hạn

while (true)

{

str = Console.ReadLine(); buff = Encoding.ASCII.GetBytes(str);

serverSocket.Send(buff, 0, buff.Length, SocketFlags.None); buff = new byte[1024];

byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive);

Console.WriteLine(str); }

## III.6. Thêm khả năng nhận dữ liệu từ client gởi lên và gởi trả dữ liệu ngược lại client

### III.6.1. Hướng dẫn lập trình

Việc này được thực hiện bằng cách thêm vòng lặp vô hạn

while (true)

{

buff = new byte[1024];

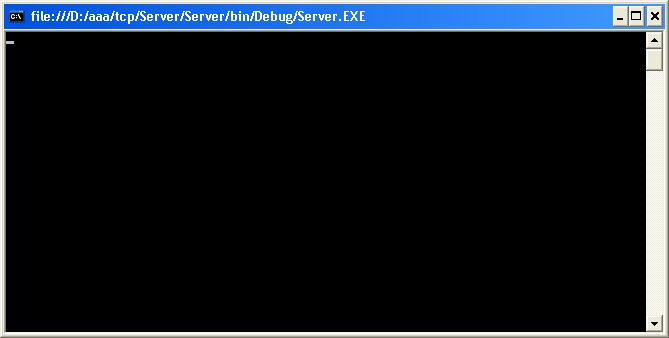
byteReceive = clientSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive); Console.WriteLine(str);

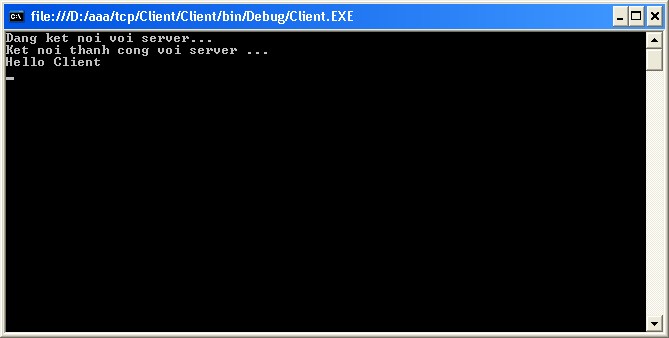
clientSocket.Send(buff, 0, byteReceive, SocketFlags.None); }

### III.6.2. Test chương trình

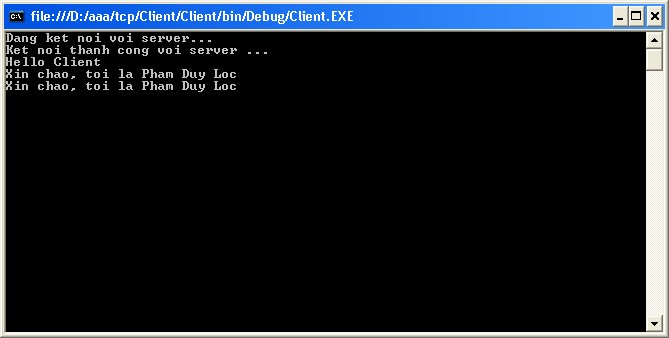
Chạy chương trình server:



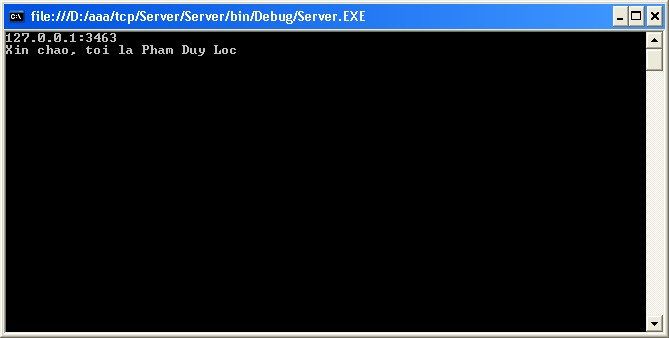
Chạy chương trình client:



Nhập dữ liệu và gởi lên server:



Kết quả hiển thị trên server:



**III.6.3. Trả lời câu hỏi:**

1. **Khi thay đổi kích thước bộ đệm từ 1024 xuống 10 thì có hiện tượng gì xảy ra ?. Giải thích**

Hiện tượng:

* Dữ liệu gửi/nhận lớn hơn 10 byte sẽ bị chia nhỏ thành nhiều gói, mỗi lần chỉ nhận được tối đa 10 byte.
* Kết quả là chuỗi nhận về sẽ bị cắt khúc, không liền mạch trong một lần Receive().
* Nếu không xử lý ghép chuỗi thì dữ liệu in ra sẽ bị thiếu hoặc hiển thị không đầy đủ.

Giải thích:

* Phương thức Receive() chỉ đọc tối đa số byte bằng kích thước bộ đệm.
* Khi buffer quá nhỏ, dữ liệu phải chia ra nhiều lần đọc 🡪 cần vòng lặp ghép lại mới có chuỗi hoàn chỉnh.
* Với buffer 1024, hầu hết thông điệp nhỏ sẽ được nhận đủ ngay trong một lần.

1. **Nếu ta không reset lại bộ đệm thì có hiện tượng gì xảy ra? Giải thích.**

Hiện tượng:

* Dữ liệu cũ trong buffer sẽ không bị xóa, nên nếu thông điệp mới ngắn hơn thông điệp cũ, phần dư sẽ vẫn còn bị in ra 🡪 gây hiện tượng chuỗi bị “rác” ở cuối.

Giải thích:

* Bộ đệm (mảng byte[] buff) là vùng nhớ tĩnh, dữ liệu cũ vẫn còn nếu không được gán lại.
* Khi Receive() trả về số byte nhỏ hơn chiều dài buffer, thì chỉ số byte đó là dữ liệu mới, phần còn lại giữ nguyên dữ liệu cũ.
* Vì vậy cần reset lại (tạo mới buff = new byte[…] hoặc Array.Clear) trước mỗi lần nhận để tránh in dữ liệu cũ lẫn vào.

# IV. Bài tập

1. Chương trình trên khi tắt client thì trên server sẽ bị lỗi, hãy cải tiến lại chương để khi client tắt đột ngột trên server sẽ thông báo client đã ngắt kết nối
2. Cải tiến chương trình: khi client nhập chữ “exit” thì sẽ tắt client, trên server thông báo client đã ngắt kết nối
3. Viết chương trình thực hiện các phép tính số học, việc thực hiện tính toán được thực hiện trên server, phía client sẽ nhập vào các toán hạng và các phép tính